

(19)



Europäisches Patentamt

European Patent Office

Office européen des brevets



REF AM

(11) EP 0 817 016 A2

(12)

EUROPEAN PATENT APPLICATION

(43) Date of publication:
07.01.1998 Bulletin 1998/02

(51) Int. Cl. 6: G06F 9/46

(21) Application number: 97110674.5

(22) Date of filing: 30.06.1997

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LU MC
NL PT SE

(30) Priority: 03.07.1996 US 675846

(71) Applicant:
SIEMENS AKTIENGESELLSCHAFT
80333 München (DE)

(72) Inventors:

- Dorn, Karlheinz, Dipl.-Inf.
90562 Kalchreuth (DE)
- Becker, Detlef, Dipl.-Ing.
91096 Mönchdorf (DE)
- Queth, Dietrich, Dipl.-Ing.
91052 Erlangen (DE)
- Reinfelder, Hans-Erich, Dr.
91054 Erlangen (DE)

(54) Software ICS for high level application frameworks

(57) An object oriented computing system, comprising objects with semantics; dynamically linkable inputs and outputs; and an event communication framework providing automated, pattern-based, fully distributable events.

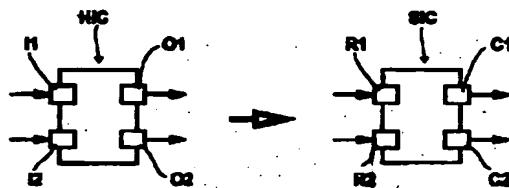


FIG 1

EP 0 817 016 A2

BEST AVAILABLE COPY

Description

BACKGROUND OF THE INVENTION

5 The present invention generally is directed to object oriented multi-programming systems. More, particularly, the invention is directed to methods and means for interconnecting software modules or building blocks.

10 As set forth in U.S. Patent No. 5,499,365, full incorporated herein by reference, object oriented programming systems and processes, also referred to as "object oriented computing environments," have been the subject of much investigation and interest. As is well known to those having skill in the art, object oriented programming systems are composed of a large number of "objects." An object is a data structure, also referred to as a "frame," and a set of operations or functions, also referred to as "methods," that can access that data structure. The frame may have "slots," each of which contains an "attribute" of the data in the slot. The attribute may be a primitive (such as an integer or string) or an object reference which is a pointer to another object. Objects having identical data structures and common behavior can be grouped together into, and collectively identified as a "class."

15 Each defined class of objects will usually be manifested in a number of "instances". Each instance contains the particular data structure for a particular example of the object. In an object oriented computing environment, the data is processed by requesting an object to perform one of its methods by sending the object a "message". The receiving object responds to the message by choosing the method that implements the message name, executing this method on the named instance, and returning control to the calling high level routine along with the results of the method. The 20 relationships between classes, objects and instances traditionally have been established during "build time" or generation of the object oriented computing environment, i.e., prior to "run time" or execution of the object oriented computing environment.

25 In addition to the relationships between classes, objects and instances identified above, inheritance relationships also exist between two or more classes such that a first class may be considered a "parent" of a second class and the second class may be considered a "child" of the first class. In other words, the first class is an ancestor of the second class and the second class is a descendant of the first class, such that the second class (i.e., the descendant) is said to inherit from the first class (i.e., the ancestor). The data structure of the child class includes all of the attributes of the parent class.

30 Object oriented systems have heretofore recognized "versions" of objects. A version of an object is the same data as the object at a different point in time. For example, an object which relates to a "work in progress", is a separate version of the same object data which relates to a completed and approved work. Many applications also require historical records of data as it existed at various points in time. Thus, different versions of an object are required.

35 Two articles providing further general background are E.W. Dijkstra, The Structure of "THE" Multi programming System, Communications of the ACM, Vol. 11, No. 5, May 1968, pp. 341-346, and C.A.R. Hoare, Monitors: Operating Systems Structuring Concepts, Communications of the ACM, Vol. 17, No. 10, October, 1974, pp. 549-557, both of which are incorporated herein by reference. The earlier article describes methods for synchronizing using primitives and explains the use of semaphores while the latter article develops Brinch-Hansen's concept of a monitor as a method of structuring an operating system. In particular, the Hoare article introduces a form of synchronization for processes and describes a possible method of implementation in terms of semaphores and gives a proof rule as well as illustrative 40 examples.

45 As set forth in the Hoare article, a primary aim of an operating system is to share a computer installation among many programs making unpredictable demands upon its resources. A primary task of the designer is, therefore, to design a resource allocation with scheduling algorithms for resources of various kinds (for example, main store, drum store, magnetic tape handlers, consoles). In order to simplify this task, the programmer tries to construct separate schedules for each class of resources. Each scheduler then consists of a certain amount of local administrative data, together with some procedures and functions which are called by programs wishing to acquire and release resources. Such a collection of associated data and procedures is known as a monitor.

50 The adaptive communication environment (ACE) is an object-oriented type of network programming system developed by Douglas C. Schmidt, an Assistant Professor with the Department of Computer Science, School of Engineering and Applied Science, Washington University. ACE encapsulates user level units and WIN32 (Windows NT and Windows 95) OS mechanisms via type-secured, efficient and object-oriented interfaces:

- 55 • IPC mechanisms - Internet-domain and UNIX-domain sockets, TLI, Named pipes (for UNIX and Win 32) and STREAM pipes;
- Event multiplexing - via select() and poll() on UNIX and WaitForMultipleObjects on Win 32;
- Solaris threads, POSIX Pthreads, and Win-32 threads;
- Explicit dynamic linking facilities - e.g., dlopen/dlsym/dlclose on UNIX and LoadLibrary/GetProcAddress on Win 32;
- Memory-mapped files;

- System VIPC - shared memory, semaphores, message queues; and
- Sun RPC (GNU rpc++).

In addition, ACE contains a number of higher-level class categories and network programming frameworks to integrate and enhance the lower-level C++ wrappers. The higher-level components in ACE support the dynamic configuration of concurrent network daemons composed of application services. ACE is currently being used in a number of commercial products including ATM signaling software products, PBX monitoring applications, network management and general gateway communication for mobile communications systems and enterprise-wide distributed medical systems. A wealth of information and documentation regarding ACE is available on the worldwide web at the following universal resource locator:

<http://www.cs.wustl.edu/~schmidt/ACE-overview.html>

The following abbreviations are or may be utilized in this application:

- **Thread** - a parallel execution unit within a process. A monitor synchronizes, by forced sequentialization, the parallel access of several simultaneously running Threads, which all call up functions of one object that are protected through a monitor.
- **Synchronizations-Primitive** - a means of the operating system for reciprocal justification of parallel activities.
- **Semaphore** - a Synchronizations-Primitive for parallel activities.
- **Mutex** - a special Synchronizations-Primitive for parallel activities, for mutual exclusion purposes, it includes a critical code range.
- **Condition Queue** - an event waiting queue for parallel activities referring to a certain condition.
- **Gate Lock** - a syntax of the monitor for each entry-function, for protection of an object, for allowing only one parallel activity at a time to use an Entry-Routine of the object.
- **Long Term Scheduling** - longtime delay of one parallel activity within a condition queue or event waiting queue for parallel activities.
- **Broker** - a distributor.

In addition, the following acronyms are or may be used herein:

| | |
|--------|---|
| AFM | Asynchronous Function Manager |
| SESAM | Services Event Synchronous Asynchronous Manager |
| PAL | Programmable Area Logic |
| API | Application Programmers Interface |
| IDL | Interface Definition Language |
| ATOMIC | Asynchronous Transport Optimizing observer-pattern-like system supporting several Modes (client/server - push/pull) for an IDL-less Communication subsystem (This is the subject of commonly assigned and co-pending application, Attorney Docket No. GR 96 P 3108 E) |
| XDR | External Data Representation |
| IO | Input/Output |
| IPC | Inter Process Communication |
| CSA | Common Software Architecture (a Siemens AG computing system conversion) |
| SW | Software |

In the past, interface of software modules or building blocks has been hard coded in an application program interface (API). This solution was linkable into a process, but was not location transparent. Additionally, the interface has been provided by way of an interface definition language (IDL) with hard coded object references.

SUMMARY OF THE INVENTION

The present invention provides a method and/or means for combining independent, semantic-less software modules or building blocks into large applications, without changing any code within the building blocks and without writing any adapters. To that end, the functionality of a module or building block is fully separated from the surrounding environment, so that it need not be located within the same process, and can instead be distributed over a network, without the need of an interface definition language.

In an embodiment, the invention provides a method for designing software modules comprising the steps of:

- defining input and output events that are fully distributable;
- configuring dynamic linkable, semantic-free software modules by input and output connections points;

providing autorouted pattern based fully distributable events based on an event communication framework.

In an embodiment, the invention provides an object oriented computing system, comprising objects with semantic-less, dynamically linkable inputs and outputs; and an event communication framework providing automated, pattern-based, fully distributable events.

In an embodiment, the inputs and outputs of the objects are provided by links to CsaConnectable and CsaRemote objects, respectively.

In an embodiment, each data structure associated with the inputs and outputs is described in a separate header file which can be used by every object to be linked.

In an embodiment, each object is a shared library which is dynamically linkable at runtime by an ASCII configuration filing names of the inputs and outputs of the objects.

These and other features of the invention are discussed in greater detail below in the following detailed description of the presently preferred embodiments with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a comparison of hardware and software ICs.

Figure 2 illustrates a application utilizing software ICs.

Figure 3 illustrates a comparison of hardware PALs and software PALs.

CO-PENDING APPLICATIONS

The following commonly assigned copending applications are incorporated herein by reference:

| Title | Application NUMBER | Filing Date | Attorney Docket No. |
|---|--------------------|-------------|---------------------|
| MONITOR SYSTEM FOR SYNCHRONIZATION OF THREADS WITHIN A SINGLE PROCESS | | | GR 96 P 3106 E |
| SERVICE AND EVENT SYNCHRONOUS/ASYNCHRONOUS MANAGER | | | GR 96 P 3107 E |
| ASYNCHRONOUS TRANSPORT OPTIMIZING OBSERVER-PATTERN LIKE SYSTEM SUPPORTING SEVERAL MODES FOR AN INTERFACE DEFINITION LANGUAGE-LESS COMMUNICATION SUBSYSTEM | | | GR 96 P 3108 E |

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EMBODIMENTS

As set forth above, the present invention provides an approach for combining independent, semantic-less building blocks into larger applications without changing any code within the building blocks and without writing any adapters.

The result is a system or architecture wherein software blocks can be combined in the same manner that integrated circuits are combinable.

For the purposes of this invention, the way a software building block connects to its outside environment (which may consist of one or more other building blocks as well as user written code) is via CsaConnectable (supplier) and CsaRemote (consumer) objects/classes, regardless whether the other endpoint of the connection resides in the same process or on a remote host (with optimization, if local). This rule is applied to all building blocks. For further information regarding the CsaConnectable and CsaRemote objects/classes, reference should be made to the commonly assigned and copending applications incorporated by reference above, and in particular Attorney Docket Nos. GR 96 P 3107 E and GR 96 P 3108 E, respectively.

Each data structure associated with the inputs and outputs of the building blocks is described in a separate header file which can be used by every building block that is to be connected.

Each building block is realized or constructed as a shared library which is dynamically linked at runtime by an ASCII configuration file as is used in public domain communication packages. The names of the inputs and outputs are assigned during dynamic linking and this allows changing the configuration without any recompilation or relinking.

The great advantage of this approach is that a flexible and high level pattern arises from the optimal combination of other simple, well designed, and semantic-less patterns. Again, this mechanism is very similar to combining integrated circuits on boards in the hardware world.

Figure 1 is useful for comparing the similarities between hardware integrated circuits (ICs) and the present software objects. In Figure 1, a hardware IC HIC has two input pins I1 and I2 and two output pins O1 and O2. Similarly, the software object SIC has two inputs R1 and R2 via CsaRemote and two outputs C1 and C2 via CsaConnectable.

An example of coding for implementing such a software IC system follows:

I. INPUT/OUTPUT CLASS DECLARATIONS

```
#ifndef SAMPLECLASS1H
#define SAMPLECLASS1H
***** */
*           *
*   Input/Output data structure *
*           *
\***** */
struct SampleClass1 {
    int      theInteger;
    DECLARE_MSC (SampleClass1)
};

IMPLEMENT_MSC (SampleClass1, V(theInteger) )

#endif // SAMPLECLASS1H
#ifndef SAMPLECLASS2H
#define SAMPLECLASS2H
***** */
*           *
*   Input/Output data structure *
*           *
\***** */
struct SampleClass2 {
    int      theInteger;
    DECLARE_MSC (SampleClass2)
};

IMPLEMENT_MSC (SampleClass2, V(theInteger) )

#endif // SAMPLECLASS2H
```

II. BUILDING BLOCK HEADER FILE

```
5  #include<ace/Service_Object.h>
6  #include<CsaConnectable.hh>
7  #include<CsaRemote.hh>
8  #include<SampleClass1.h>
9  #include<SampleClass2.h>
10 class SampleApplication : public ACE_Service_Object
11 {
12     public:
13         virtual int init (int, char**);
14         virtual int fini (void);
15         virtual int info (char**, size_t) const;
16         SampleApplication (void);
17         ~SampleApplication (void);
18     protected:
19         CsaConnectable <SampleClass1> *output1;
20         CsaConnectable <SampleClass2> *output2;
21         CsaRemote <SampleClass1> *input1;
22         CsaRemote <SampleClass2> *input2;
23     };
24
25 #endif/ / SAMPLE_APPLICATION
```

III. BUILDING BLOCK IMPLEMENTATION

```

5 #include<CsaConnectable.hh>
6 #include<CsaRemote.hh>
7 #include<SampleApplication.h>
8
9 int SampleApplication :: init(int argc, char **argv) {
10     cout << endl << "Initializing" << endl;
11     input1 = new CsaRemote <SampleClass1> (argv[1]);
12     input2 = new CsaRemote <SampleClass2> (argv[2]);
13     output1 = new CsaConnectable <SampleClass1> (argv[3]);
14     output2 = new CsaConnectable <SampleClass2> (argv[4]);
15     return (0);
16 }
17
18 int SampleApplication :: fini (void) {
19     cout << endl << "Finalizing" << endl << endl;
20     delete input1;
21     delete input2;
22     delete output1;
23     delete input1;
24     return (0);
25 }
26
27 int SampleApplication :: info(char**, unsigned) const {
28     cout << endl << "Returning infos about" << endl;
29     return (0);
30 }
31
32 SampleApplication :: SampleApplictation(void) {}
33 SampleApplication :: ~SampleApplication(void) {}
34 /* Dynamically linked functions used to control configuration */
35 extern "C" ACE_Service_Object *_alloc(void);
36 ACE_Service_Object *_alloc (void) {
37     return (ACE_Service_Object *)new SampleApplication;
38 }
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
988
989
989
990
991
992
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2289
2290
2291
2292
2293
2294
2295
2295
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2389
2390
2391
2392
2393
2394
2395
2395
2396
2397
2398
2399
2399
2400
2401
2402

```

IV. ASCII CONFIGURATION FILE

```

5     static SVC_Manager      "-d -p 3333"
dynamic SampleApplication
10    ACE_Service_Object * ./SampleApplication.so:_alloc()
     "SampleApplication in1_name in2_name out1_name out2_name"

```

15 In Figure 2 there is illustrated in block diagram form a possible implementation of software ICs in a system with
more than one application. In Figure 2 there are illustrated five software ICs: IC1, IC2, IC3, IC4 and IC5. Additionally,
there are two applications, Application 1 and Application 2, employing the software ICs. Application 1 contains software
ICs IC1, IC2 and IC3, while Application 2 contains software ICs IC4 and IC5. As can be seen, Application 1 and Appli-
cation 2 interact with each other, as well as externally of the process or system containing Application 1 and Application
2, via inputs and outputs of the software ICs.

20 As illustrated, IC1 has two inputs C11 and C12. IC1 also has one output via R11. The inputs C11 and C12 are connected
to two outputs of IC2, R21 and R22, respectively. An input C21 of IC2 is connected to the output R11 of IC1.

25 IC3 has an output R31 connected to the input C22 of IC2, and input C31 connected externally of the process con-
taining the applications, an input C32 connected to an output R41 of IC4 and an output R52 connected to an input C52 of
IC5 and externally of the system. In addition to output R41, IC4 has a input C41 connected externally of the system
and an output R42 connected to an input C51 of the IC5. IC5 also has an output R51 connected externally of the proc-
cess or system containing the applications.

30 The inputs and output are via CsaConnectable and CsaRemote as described above. Moreover, the data are auto-
souted to the various inputs and outputs via dynamic linking, thereby allowing changing the configuration and interaction
of the applications without requiring recompilation or relinking.

35 In addition, the foregoing software IC principles can be combined with a pattern (task) from ACE, to obtain a very
powerful software building block that behaves like a hardware PAL, and that offers the power of synchronous behavior
within the building block and asynchronous behavior/interaction outside

40 of the building block. The internal processing rate (the counterpart to a clock rate in a hardware PAL) is thus fully
independent from the event input/output rate of the connected environment. The necessary buffering to achieve the
synchronization is also provided without concern to semantics. Similar to hardware PAL synchronization solutions, the
synchronization task can be configured into a software PAL, as needed.

45 Figure 3 illustrates a comparison between hardware PALS and Software PALS. As illustrated, a hardware PAL 310,
like a hardware IC, can have two input pins I1 and I2 and two output pins O1 and O2. However, within the hardware PAL
310 there also are provided registers/buffers reg in which incoming and outgoing data or signals are stored.

50 The counterpart software PAL 312 has inputs R1 and R2 and outputs C1 and C2 like the software IC described pre-
viously. However, also illustrated are tasks T1 and T2 that replace the registers/buffers reg of the hardware PAL 310. In
other respects, the software PAL is similar to a software IC, as described above.

55 A software PAL provides inner logic flexibility to a software IC by means of active object support. Incoming events
are able to be buffered by tasks, such as the task T1, before further processing by the inner logic. Further, outgoing
events can be taken from a buffer, such as the task T2, thereby decoupling the events from the inner logic of the soft-
ware PAL.

60 Although modifications and changes may be suggested by those skilled in the art, it is the intention of the inventors
to embody within the patent warranted hereon all changes and modifications as reasonably and properly come within
the scope of their contribution to the art.

Claims

1. An object oriented computing system on a computer platform, comprising:
55 objects with semanticless, dynamically linkable inputs and outputs; and
an event communication framework providing automated, pattern-based, fully distributable events.
2. The object oriented computing system according to claim 1, wherein the inputs and outputs of the objects are pro-

vided via CsaConnectable and CsaRemote objects, respectively.

3. The object oriented computing system according to claim 2, wherein each data structure associated with the inputs and outputs is described in a separate header file which can be used by every object to be linked.
4. The object oriented computing system according to claim 2 or 3, wherein each object is a shared library which is dynamically linkable at runtime by an ASCII configuration filing names of the inputs and outputs of the objects.
5. An object oriented computing system on a computing system, comprising objects having dynamically linkable inputs and outputs and internal tasks for queuing of data transferred into and out from the objects via said inputs and outputs, respectively; and an event communication framework providing automated, pattern-based, fully distributable events.
6. The object oriented computing system according to claim 5, wherein the inputs and outputs of the objects are provided via CsaConnectable and CsaRemote objects, respectively.
7. The object oriented computing system according to claim 6, wherein each data structure associated with the inputs and outputs is described in a separate header file which can be used by every object to be linked.
8. The object oriented computing system according to claim 6 or 7, wherein each object is a shared library which is dynamically linkable at runtime by an ASCII configuration file containing names of the inputs and outputs of the objects.
9. A method for designing software modules in an object oriented computing system, comprising the steps of:
 - defining input and output events that are fully distributable;
 - configuring dynamic linkable, semantic-free software modules by input and output connections points;
 - providing automated pattern based fully distributable events based on an event communication framework.
10. A storage medium including object oriented code having an object oriented computing system on a computer platform, comprising:
 - objects with semanticless, dynamically linkable inputs and outputs; and
 - an event communication framework providing automated, pattern-based, fully distributable events.
11. The storage medium according to claim 10, wherein the inputs and outputs of the objects are provided via CsaConnectable and CsaRemote objects, respectively.
12. The storage medium according to claim 11, wherein each data structure associated with the inputs and outputs is described in a separate header file which can be used by every object to be linked.
13. The storage medium according to claim 11 or 12, wherein each object is a shared library which is dynamically linkable at runtime by an ASCII configuration filing names of the inputs and outputs of the objects.
14. A storage medium including object oriented code for an object oriented computing system on a computing system, comprising objects having dynamically linkable inputs and outputs and internal tasks for queuing of data transferred into and out from the objects via said inputs and outputs, respectively; and an event communication framework providing automated, pattern-based, fully distributable events.
15. The storage medium according to claim 14, wherein the inputs and outputs of the objects are provided via CsaConnectable and CsaRemote objects, respectively.
16. The storage medium according to claim 15, wherein each data structure associated with the inputs and outputs is described in a separate header file which can be used by every object to be linked.
17. The storage medium according to claim 15 or 16, wherein each object is a shared library which is dynamically linkable at runtime by an ASCII configuration file containing names of the inputs and outputs of the objects.

18. A storage medium including object oriented code for a method for designing software modules in an object oriented computing system, comprising the steps of:

5 defining input and output events that are fully distributable;
configuring dynamic linkable, semantic-free software modules by input and output connections points;
providing autorouted pattern based fully distributable events based on an event communication framework.

10

15

20

25

30

35

40

45

50

55

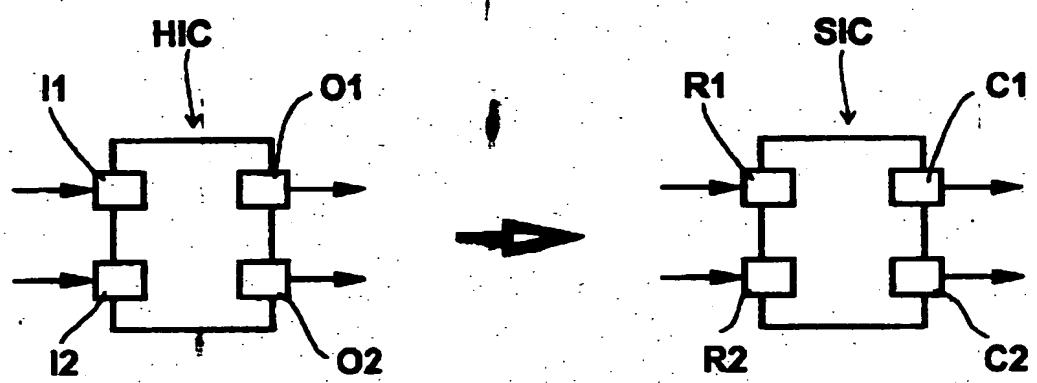


FIG 1

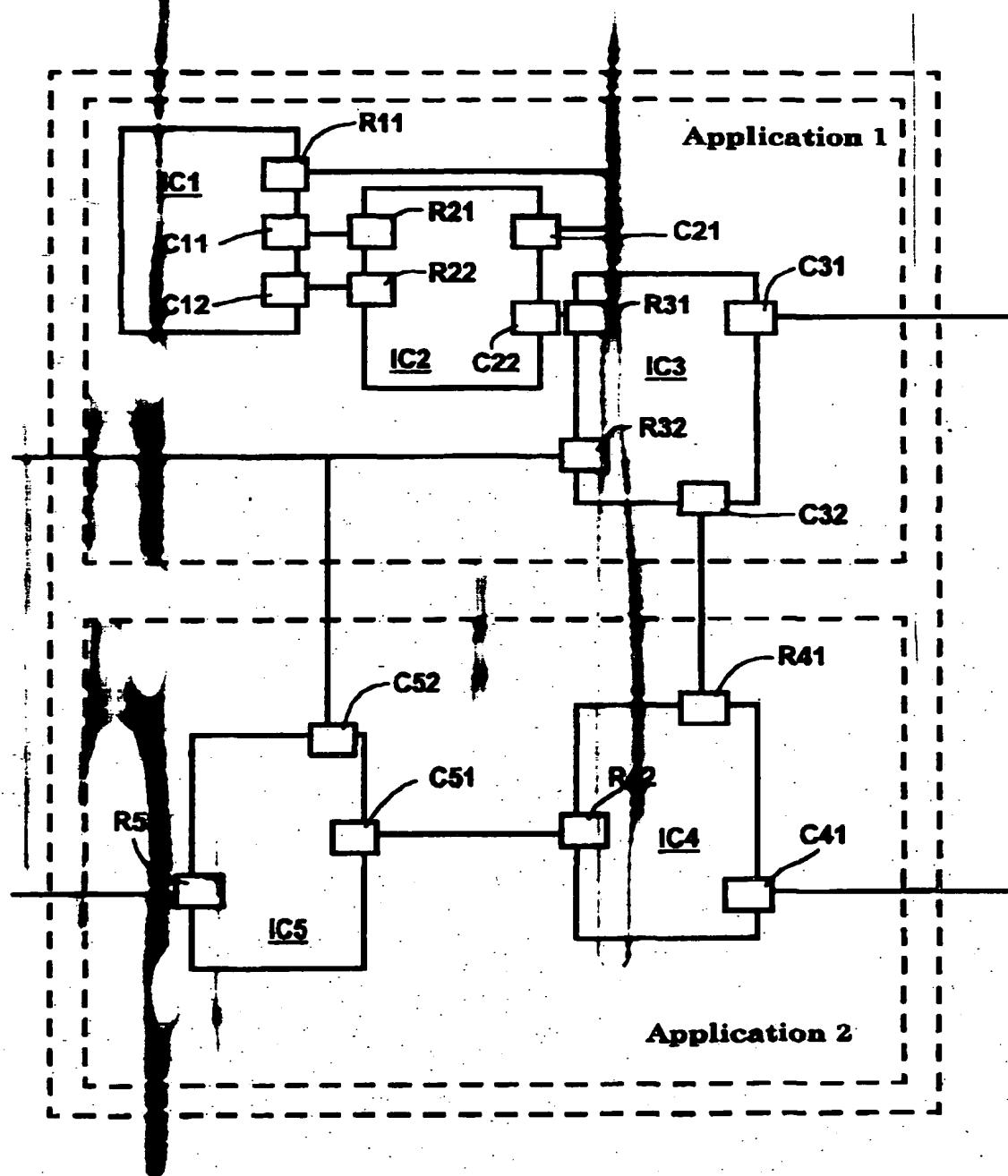


FIG 2

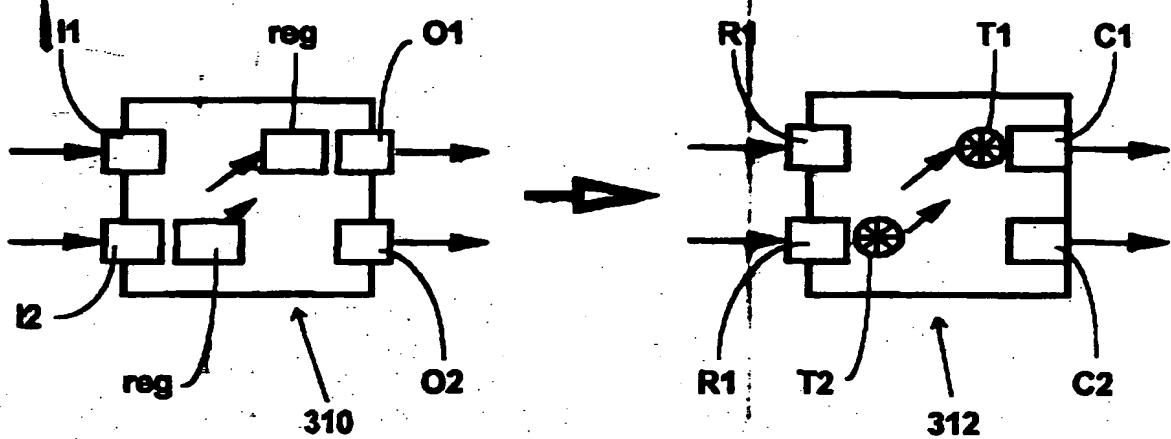


FIG 3

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- BLACK BORDERS**
- IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- FADED TEXT OR DRAWING**
- BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- SKEWED/SLANTED IMAGES**
- COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- GRAY SCALE DOCUMENTS**
- LINES OR MARKS ON ORIGINAL DOCUMENT**
- REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.